ANL/MCS-TM-316

Report on FY11 Extensions to MeshKit and RGG

Mathematics and Computer Science Division

ANL/MCS-TM-316

Report on FY11 Extensions to MeshKit and RGG

prepared by
Rajeev Jain
Mathematics and Computer Science Division, Argonne National Laboratory

September, 30, 2011

Introduction

One of the barriers to performing high-fidelity computational simulation of reactor core phenomena is the production of good-quality geometry and mesh models required by these simulations. Although a variety of geometry and meshing tools are available, they suffer from shortcomings in usability, robustness, or generality which makes them difficult to apply to reactor applications. The SHARP frameworks project is addressing these deficiencies by developing a library of mesh generation algorithms, and tools based on that library. The library is known as MeshKit, and one of the tools being developed is named RGG, for Reactor Geometry (and mesh) Generator.

Over the past year, our meshing-related work was split between general design improvements in MeshKit, enhancing RGG to run in parallel, and the application of RGG to several specific reactor designs. These activities are described in this report.

2 MeshKit Design

Early development activities in MeshKit focused on adding raw meshing-related capabilities. Capabilities were developed for surface (tri, quad) and volume (tet, hex) mesh generation, for interacting with geometric models, and various other infrastructure used in the meshing process (e.g. mesh smoothing, mesh-based geometry). Without a unifying design, the various algorithms in MeshKit could not be used together without extra effort. During FY11, we developed and implemented a unified MeshKit design which facilitates interactions between the MeshKit algorithms.

The unifying theme behind this new design for MeshKit is that the meshing process can be posed as a graph-based problem, with nodes of the graph representing a meshing-related function and edges representing data dependencies between nodes. In a traditional geometry- based meshing approach, the nodes might correspond to geometric vertices, edges, faces, and regions. However, other mesh generation approaches can also be described by graphs, with graph nodes representing other data concepts. For example, a copy/move/merge meshing approach, like that supported by RGG, with nodes for generating the geometry, then the mesh, for the different assembly types, nodes for copying assembly types into the core lattice, and a single node for merging coincident vertices. After implementation of the graph-based design into core MeshKit classes, the various algorithms and tools in MeshKit were moved into the new design. After these changes, interactions between the tools were much easier to implement. A complete description of the new MeshKit design is beyond the scope of this report, but can be found in Ref. [1].

In FY11 we also developed a new, open-source all-quadrilateral meshing algorithm Jaal. This method is based on combining triangles to form quadrilaterals, and has demonstrated good robustness for a variety of geometric models. This algorithm is described in a forthcoming paper at the International Meshing Roundtable conference [2].

3 RGG Enhancements

AssyGen was implemented in the new graph-based meshing approach, currently under development in MeshKit. In the graph-based design after AssyGen operation, various open source meshers such as triangle, CAMAL tet-mesher, Jaal quad-mesher and parallel tet-

mesher can be used to mesh the assembly geometry. The user interface and robustness of the library-based meshing are still under development. Current work involves implementing CoreGen in the new MeshKit design. Several other enhancements were made to RGG, based either on user requests or deficiencies identified in the applications described in the next section of this report:

- **CoreGen support for geometric models:** CoreGen was modified to allow generation of geometric models of a core lattice, instead of just mesh models. The user input for generating core geometry models is almost identical to that used for mesh models; the new "ProblemType" keyword is used to specify if geometry or mesh is desired.

- **Support for ACIS or OCC:** RGG was used to further debug the port of the Common Geometry Module (CGM) to the Open.CASCADE solid modeling engine. AssyGen and CoreGen now support generation of models using either OCC or ACIS, depending on how the version of CGM used by RGG is configured.

New keywords: New keywords "EdgeInterval" and "CreateSideSet" were introduced in the AssyGen scripting language for specifying the interval on outer edges of the assembly geometry and for specifying controls for sidesets creation, respectively. The "CreateNeumannSet" keyword for CoreGen can be used to specify Neumann sets on top, side, and/or bottom sides of the overall core mesh.

Earlier in the year, the capability was developed in MOAB to perform vertex merging in parallel, using a small modification to the algorithm which matches shared inter-processor interfaces in the mesh [3]. This capability also enabled the development of the parallel CoreGen tool. The basic algorithm used in this tool can be summarized in five steps:

1. On each processor: read CoreGen input file, parse, and determine assembly copies assigned to this processor based on a round-robin distribution.

2. Locally read assembly meshes for assemblies determined in step 1.

3. Perform assembly copy/move operations assigned to this processor.

4. Perform parallel merge.

5. Save output mesh.

The performance of parallel CoreGen was measured using a 1/6 VHTR core model consisting of 56 assemblies. Meshes of 11M and 58M hexes were generated, on up to 56 processors of the ANL Fusion cluster computer. Table 1 summarizes the model details of this model.

Table 2 shows the maximum value (among all processors) of wall clock time, memory used, time to - load mesh files, copy/move mesh files, merge coincident nodes and save mesh. It is observed that as the number of processors increases and the work gets distributed the memory and time requirement for each sub-process decreases. The low speedup obtained when using 8 processors is due to higher communication cost during the merge operation and the fact that some processors have more than one mesh file loaded which causes load imbalance. This imbalance is resolved when the number of processors is greater than nA.

Table 1. VHTR 1/6 core model details. nA and nT are the number of assembly mesh files and total number of assemblies forming the core, respectively.

#Elements, Volumes	58.9 M hexes, 5536
#nA, nT	12, 56
#Interval Axial (Z), Radial Direction	80, 72
Core Mesh File Size (GB)	6.52

Table 2. VHTR 1/6 core time and memory results

Procs	Walltime (mins)	Max. CPU time (mins)	Max. Clock time (mins)	Max. Memory used (GB)	Max. Load mesh time (mins)	Max. Copy/ move time (mins)	Max. Merge time (mins)	Max. Save time (mins)
1	215.3	214.8	215	14.1	0.25	141.8	70.48	2.25
8	181.9	89.8	181.9	7.4	0.1	59.6	28.29	1.69
16	20.18	19.8	20	4.9	0.018	1.4	17.62	0.8
32	2.9	2.7	2.8	1.29	0.018	0.12	2.34	0.25
56	1.23	1.01	1.11	0.84	0.018	0.001	0.8	0.18

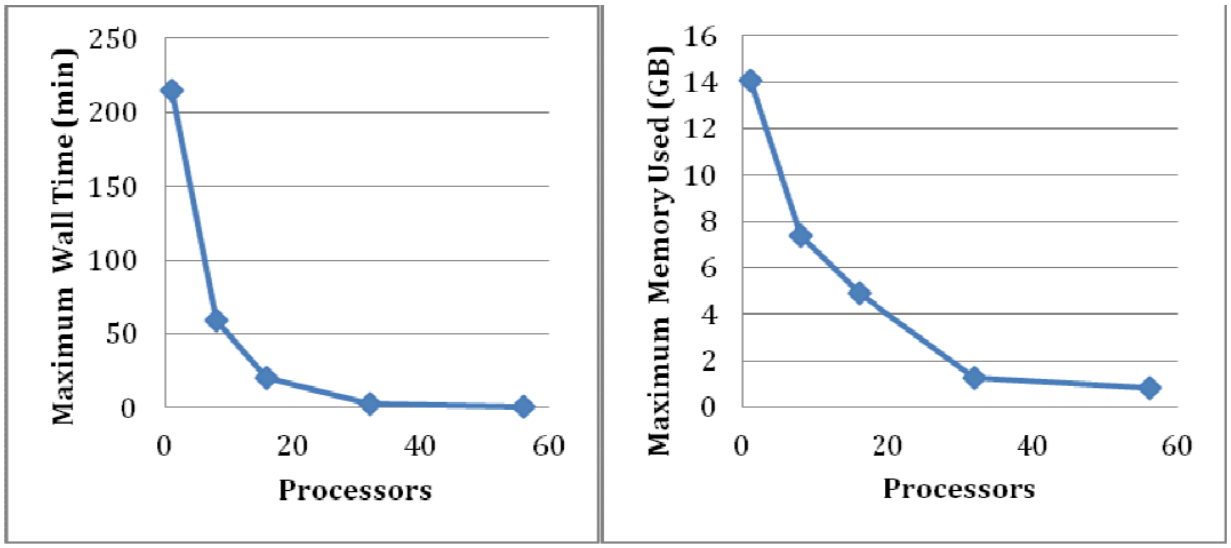

Figure 1. Plots for maximum wall time (min) and maximum memory used by a processor (GB) vs number of processors.

RGG Applications

VHTR 1/12, 1/6 and Full-Core Models

Figure 2 shows geometries of 1/12th (right) and 1/6th (left) VHTR core, picture at the center is a close-up of the mesh output from the CoreGen tool. Figure 3 shows the full core model generated using RGG tools. Moving from left to right the rectangle in red shows

zoomed view of the picture on the left. It must be noted that the blue outer covering of the assemblies is an interstice mesh which is carefully created offline to match other assemblies

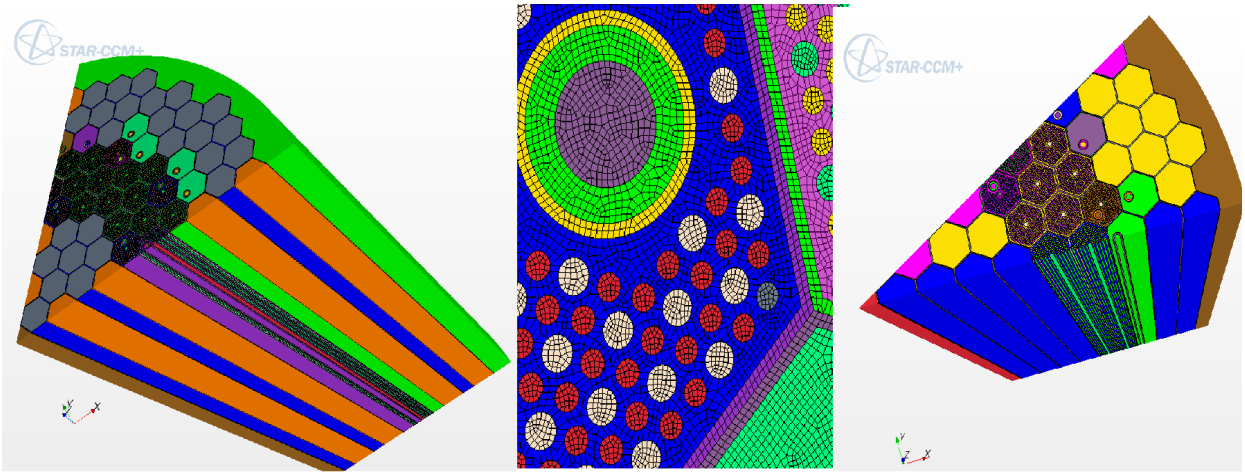

Figure. 1. VHTR 1/6th: 53M hexes takes 10.4GB RAM and 12 mins (left), closeup (center), VHTR 1/12th (right).

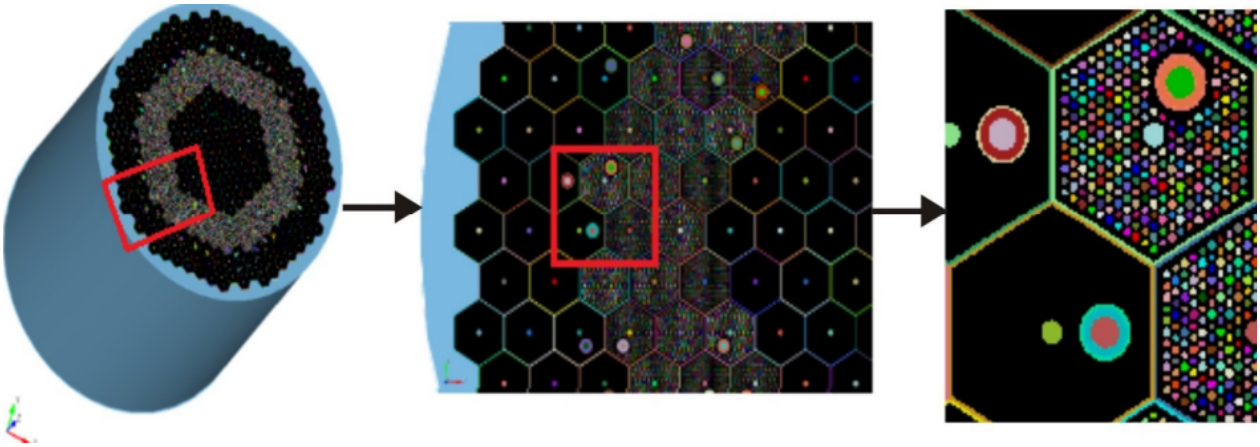

Figure 2. VHTR full-core: 23M hexes, 33k vols take 5.5 GB RAM and 30 min; 313 assemblies.

1/4th PWR Benchmark Reactor

The benchmark problem “MOX Fuel Loaded Small PWR Core” can be found on the website of the Nuclear Reactor Analysis and Particle Transport surface of the core geometry and zoomed view of three regions A, B and C. This was model is [9]. Figure 4 shows the top created using the geometry feature of CoreGen program.

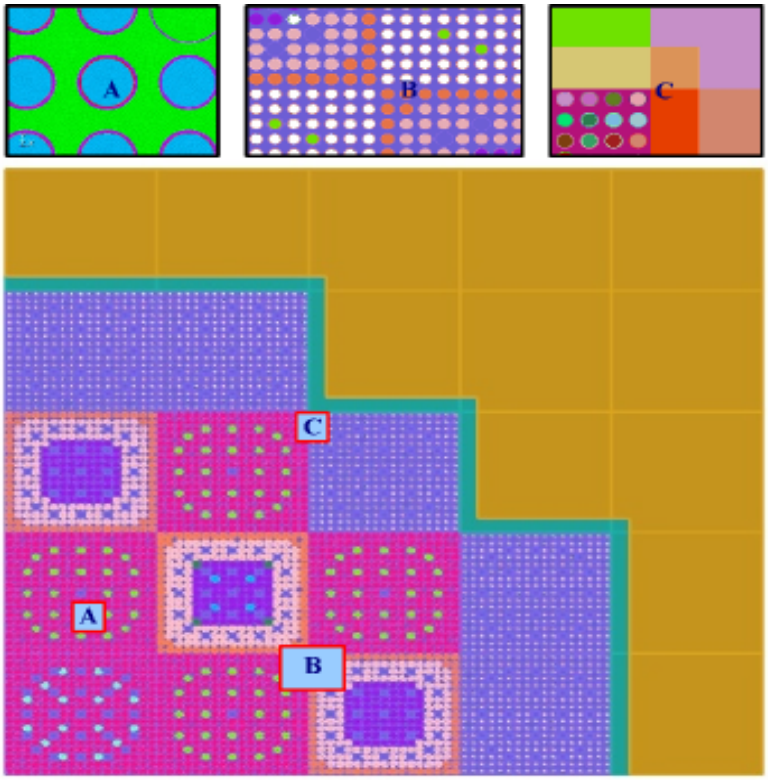

Figure 5. 2D geometry of a 1/4 PWR benchmark reactor, 11k vols takes 0.4 GB RAM and 18 mins.

Full Core MONJU Reactor

STARCCM+ was used to visualize this full-core MONJU reactor model. The model and 715 assemblies that are individually meshed using the CUBIT mesh generation toolkit. This resulting core is generated serially on a Linux workstation.

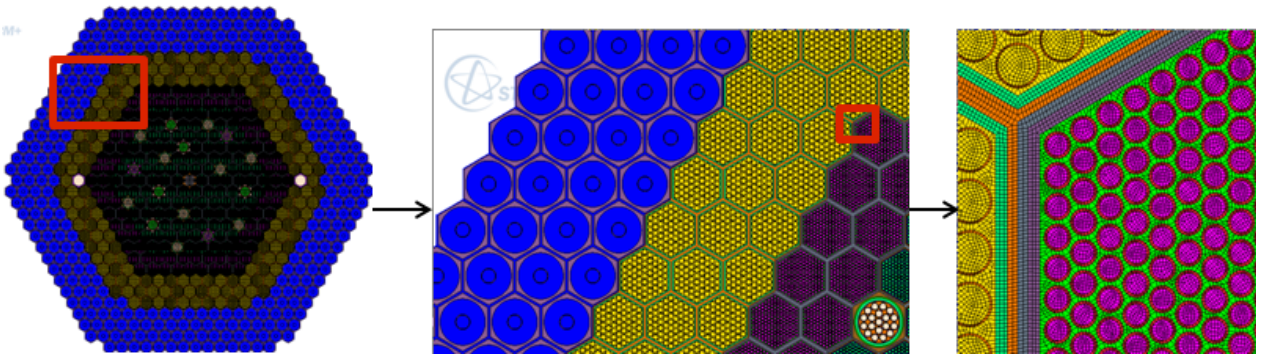

Figure. 4. MONJU reactor, full core model: 9.7M hexes, 99k vols takes 4.3GB RAM and 176 mins. 715 assemblies.

References

1. Tautges, T. J., Kraftcheck, J., Porter, Jim, Caceres, Alvaro, Grindeanu, Iulian, Karpeev, Dmitry, Jain, Rajeev, Kim, Hong-Jun, Cai, Shengyong, Jackson, Steve, Hu, Jiangtao, Smith, Brandon, Verma, Chaman, Slattery, Stuart, Wilson, Paul: MeshKit: A Open-Source Library for Mesh Generation. Proceedings, SIAM Conference on Computational Science & Engineering. SIAM, Reno, NV (2011), http://trac.mcs.anl.gov/projects/fathom/wiki/MeshKit.

2. Verma, Chaman, Tautges, TimothyJ.: Jaal: Engineering a high quality all-quadrilateral mesh generator. Presented at the International Meshing Roundtable, Paris, France October 23 (2011).

3. Tautges, Timothy, J., Kraftcheck, Jason, Bertram, Nathan, Sachdeva,Vipin, Magerlein, John: Mesh Interface Resolution and Ghost Exchange in a Parallel Mesh Representation. Presented at the 26th IEEE International Parallel & Distributed Processing Symposium, Shanghai, China. May 21 (2012).

4. Jain, Rajeev: RGG: A Tool to Generate Reactor Core Models. VHTR 4th Annual Technical Review Meeting. Albuquerque, NM (2011).

5. Jain, Rajeev: RGG:A Tool to Generate Reactor Core Models. 11th National Congress on Computational Mechanics. Minneapolis, Minnesota (2011).

6. Thomas, Justin, Jain, Rajeev: Integrating STAR-CD with Systems Analysis Code for Nuclear Reactor Safety Simulations. 2011 STAR American Conference. , Chicago, IL (2011).

7. Reactor Geometry (& Mesh) Generator (RGG), http://trac.mcs.anl.gov/projects/fathom/wiki/rgg.

8. Tautges, T.J., Jain, Rajeev: Creating Geometry and Mesh Models for Nuclear Reactor Core Geometries Using a Lattice Hierarchy-Based Approach. Engineering With Computers. to appear, (2011).

9. Benchmark Problems in Reactor and Particle Transport Physics, http://nurapt.kaist.ac.kr/benchmark/.

Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 240
Argonne, IL 60439-4847

www.anl.gov